GPU-based cross-platform Monte Carlo proton dose calculation engine in the framework of Taichi

Weiguang Li[1,2,3], Cheng Chang[3], Yao Qin[3], Zilu Wang[3], Kai-Wen Li[3], Li-Sheng Geng[1,4,5], Hao Wu[2,6]

1. *School of Physics, Beihang University, Beijing 102206, China*
2. *Key Laboratory of Carcinogenesis and Translational Research (Ministry of Education/Beijing), Department of Radiation Oncology, Peking University Cancer Hospital & Institute, 52 Fucheng Road, Beijing 100142, China*
3. *Medical Management Department, CAS Ion Medical Technology Co., Ltd., Beijing 100190, China*
4. *Beijing Key Laboratory of Advanced Nuclear Materials and Physics, Beihang University, Beijing 102206, China*
5. *School of Physics and Microelectronics, Zhengzhou University, Zhengzhou, Henan 450001, China*
6. *Institute of Medical Technology, Peking University Health Science Center, 38 Xueyuan Road, Beijing 100191, China*

Abstract

In recent years, graphics processing units (GPUs) have been applied to accelerate Monte Carlo (MC) simulations for proton dose calculation in radiotherapy. Nonetheless, current GPU platforms, such as CUDA and OpenCL, suffer from cross-platform limitation or relatively high programming barrier. However, the Taichi toolkit, which was developed to overcome these difficulties, has been successfully applied to high-performance numerical computations. Based on the class II condensed history simulation scheme with various proton-nucleus interactions, we developed a proton MC transport GPU-accelerated engine using the Taichi toolkit. Dose distributions in homogeneous and heterogeneous materials were calculated for 110, 160, and 200 MeV protons and were compared with those obtained by full MC simulations using Topas. The gamma passing rates were greater than 0.99 and 0.95 with criteria of 2 mm, 2% and 1 mm, 1%, respectively, in all the tested conditions. Moreover, the calculation speed was at least 5800 times faster than that of Topas, and the number of lines of code was approximately 10 times lesser than those of CUDA or OpenCL. Our study provides a highly accurate, efficient, and easy-to-use proton dose calculation engine for algorithm developers, students, and medical physicists.

Keywords: Proton therapy; Monte Carlo dose calculation; GPU acceleration; Taichi

1. Introduction

The availability of proton therapy has dramatically increased in recent years. According to the 2021 statistics of the ParticleTherapyCo-Operative Group(PTCOG)[1], 98 proton therapy facilities are operational worldwide, 19 facilities are under construction, and 4 facilities are being planned. Owing to the sharp falloff in the dose distribution (Bragg Peak), a proton beam can deliver most of the dose to tumors, thus protecting the surrounding healthy tissues. In addition, with the development of new types of beamline systems, such as Huazhong University of Science and Technology Proton Therapy Facility(HUST-PTF)[2] and treatment techniques, such as flash therapy[3], proton dose calculation requires much higher accuracy than that in conventional photon radiotherapy[4]. Most current commercial proton-dose calculation engines used in treatment planning systems are based on analytical algorithms which can rapidly yield accurate dose results for homogeneous tissues. However, the predicted results are inaccurate when highly heterogeneous tissues are considered [5]. In addition, proton therapy produces many secondary particles, the information on which is of great importance for analyzing their physical and biological effects [6]. The analytical algorithms fail to provide detailed information on secondary fragments.

In contrast, Monte Carlo (MC) algorithms fully simulate particle reactions and model transport geometries, achieving accurate total and fragment dose results. Thus, MC simulations are considered the gold standard in dose calculations, although they are time-consuming. Currently, widely used MC tools, such as Geant4, Gate, Topas, and FLUKA, are based on Central Processing Units (CPUs). However, Graphics Processing Units (GPUs) can integrate more computing units, which can simulate transport processes in parallel [7] [8] [9] and significantly reduce calculation time. Jia et al. developed the first GPU-based MC proton-dose calculation engine, called gPMC [10]. The physics model was adapted from Fippel and Soukop [11], where proton transport was simulated in a class II condensed history scheme with a continuous slowing down of approximations. Ionization and elastic and inelastic interactions between the protons and materials were included. Chan et al. [12] applied a more precise nonelastic nuclear reaction model that incorporated the Bertini cascade and evaporation kernels by sacrificing speed. Both models are CUDA-based, and their applications are limited to Nvidia GPUs. New versions of gPMC, gPMC v2.0, and goMC were realized in the OpenCL framework [13] [14], facilitating cross-platform applications. Although OpenCL supports multiple GPU hardware, it is mainly developed using an Application Programming Interfaces(API) with a specific hardware development interface.

In 2019, Hu et al. proposed a new parallel programming language called Taichi for high-performance numerical computations [15] [16] [17]. It supports most mainstream GPUs, and thus works well in cross-platform environments. Taichi is embedded in Python and uses just-in-time (JIT) compiler frameworks to offload computing-intensive Python code to native GPU or CPU instructions. Using the @ti.kernel decorator, Taichi's JIT compiler automatically compiles Python functions into an efficient GPU or CPU machine code for parallel execution. These features enable easy programming that significantly reduces the total number of lines of code compared to OpenCL and CUDA. Moreover, Taichi can be well integrated into mainstream Python deep learning ecosystems such as PyTorch, which integrate GPU high performance computing with deep learning neural networks [18] [19] [20]. Consequently, Taichi has been widely used in high-performance physics studies [16] and machine learning fields [21]. However, Taichi has not yet been applied to dose calculations in radiotherapy. In this work, we developed a new cross-platform GPU-based MC proton dose calculation engine for the Taichi platform, named Taichi Proton Monte Carlo(TPMC), with a model of proton transport similar to those in [11] [14], except for a different inelastic nuclear reaction model.

The strategy for GPU acceleration in Taichi is similar to those in OpenCL and CUDA. Primary and secondary stacks were built to store primary protons and secondary particles, respectively. The GPU calculation threads first processed the primary protons to start transport and then save all the secondary particles produced into the secondary stacks. After all the primary protons were stopped or escaped from the region of interest, the secondary protons were transported on the GPU threads again and treated as primary protons, except that the information of the produced secondaries was not documented. Using the Taichi platform reduced the number of lines of code to only a few hundred. The computation time was reduced by allocating the deposited dose to different local dose counters to avoid the atomic addition

effect. In our study, we prepared 110-, 160-, and 200 MeV proton pencil beams and four water and human tissue phantoms for dose calculation. The gamma passing rates, calculation speeds, and total number of lines of code were evaluated and compared with those obtained from Geant4-Topas and other GPU-based simulations.

The remainder of this study is organized as follows. In Section 2, we describe the physical process of the TPMC, parallel simulation strategy, and Taichi GPU memory arrangement. The results of the dose calculations for Taichi and the other platforms are compared in Section 3. Finally, in Section 4, we conclude our study and discuss its current and future applications.

2. Methods

In this study, we employed the class II condensed history simulation scheme [22] and continuous slowing down approximations for proton transport. A simplified physics model was established to simulate the nuclear reactions of primary and secondary protons. The physical process of proton transport was almost the same as those in Refs [10] [11], which will be briefly introduced in the following section. On the other hand, proton transport was accelerated in the Taichi framework on GPUs, where parallel threads were applied and a new dose counter model was developed to solve thread-block problems.

2.1 Physics model

Protons were transported step by step in materials. The step size is defined as

(1)

where is the maximum step size required to ensure that energy loss remains within 25% of the total kinetic energy or 2 mm [22] in each step. is the distance between the particle position and voxel surface, is a random number in the range of [0-1], andis the total reaction cross section. The step length in human tissue was first converted to its corresponding water equivalent length (with the same energy loss in water)

. (2)

Here , denote the material density, density of water, kinetic energy of the protons, and relative stopping power of the voxel material to water, which was determined in [11], respectively. At each step, the angular distribution of the scattered protons was calculated using Moliere's function [23][24]. The average proton energy loss in one step was obtained by solving

(3)

where is the restricted stopping power of water, which was calculated using the Bethe-Bloch formula [25]. To consider the effects of secondary electrons in the ionization process less than the cutoff energy = 0.1 MeV, the actual energy loss was varied with a Gaussian fluctuation , where the standard deviation was determined following Ref [23]. In addition, all secondary electrons deposited their energy, as their short

range compared to the Computed Tomography(CT) voxel size. Post-step reactions between protons and materials occurred after each step, including ionization, proton-proton elastic nuclear interactions, proton-oxygen elastic nuclear interactions, and proton-oxygen inelastic nuclear interactions. These interactions were treated in the same manner as in Ref. [10], except that the inelastic cross sections were extracted from MC simulations using Geant4[26] and tabulated as inputs to accelerate the MC simulation, as suggested by Wan et al.[12].

2.2 GPU acceleration

As mentioned in the previous Section 1, the implementation of CUDA is limited to the GPU of NVIDIA. Although OpenCL can solve this problem [13], it requires an additional application programming interface and lacks functional libraries, thereby creating higher programming barriers for medical physicists and developers. However, Taichi can overcome the drawbacks of these two languages [15]. It is based on Python and can therefore conveniently use available packages such as random, numpy, and pandas. It is to be noted that there is even no random number generator in OpenCL-based libraries [8]. When applied to GPU acceleration, an easy outermost scope for-loop command automatically parallelizes work items on different threads in the Taichi kernel, whereas in CUDA and OpenCL, appropriate GPU machine codes must be designed [15]. Moreover, switching from Nvidia cards to other GPU cards can be achieved simply by changing only the head files in Taichi, instead of (almost completely) rewriting the code in OpenCL. Finally, the compilation time and memory space are saved in Taichi because it is performed on the LLVM-based compiler Clang [15], and Taichi also tunes the parameters to best explore the GPU architecture. Table 1 briefly compares these three languages.

Table 1: Development environment and bases of different platforms.

Platform	Base	GPU	Library
CUDA	C++	Nvidia	Nvidia
OpenCL	API	Multiple	OpenCL
Taichi	Python	Multiple	Python

The cross-sectional data of proton-nucleus reactions and the density and stopping power distribution of the target were first loaded by the CPU and stored in the GPU global memory, which was called constantly during the simulations. The transport of primary protons was randomly simulated stepwise on different GPU threads until they stopped or moved out of the target. The automatic parallelization of particles on a GPU can be achieved using @ti.for-loops command in TPMC. Secondary particles were generated after inelastic nuclear interactions, and their information, including location, energy, and direction, was recorded in the GPU global memory and pushed into the secondary particle stack. The transport of secondary particles was blocked until all the primary protons were simulated. Subsequently, we randomly allocated secondary protons to different threads for transport.

Although the particles were transported in parallel on the GPUs, the dose deposition process in each thread was not independent. Individual dose deposition on the global dose counter causes a thread block, where the dose information of other threads must wait until the current process of proton information transfer ends, known as atomic addition. This treatment was the primary reason for the time-consuming nature of the GPU-based MC dose engine. In this study, we applied the same strategy as in gPMC V2.0[14] to mitigate the memory conflict problem. Eight local dose counters were prepared to balance the calculation efficiency and memory size. The deposited energy was randomly assigned to a dose counter after each step, and the total doses were summed for the eight counters after all the particle transports were completed.

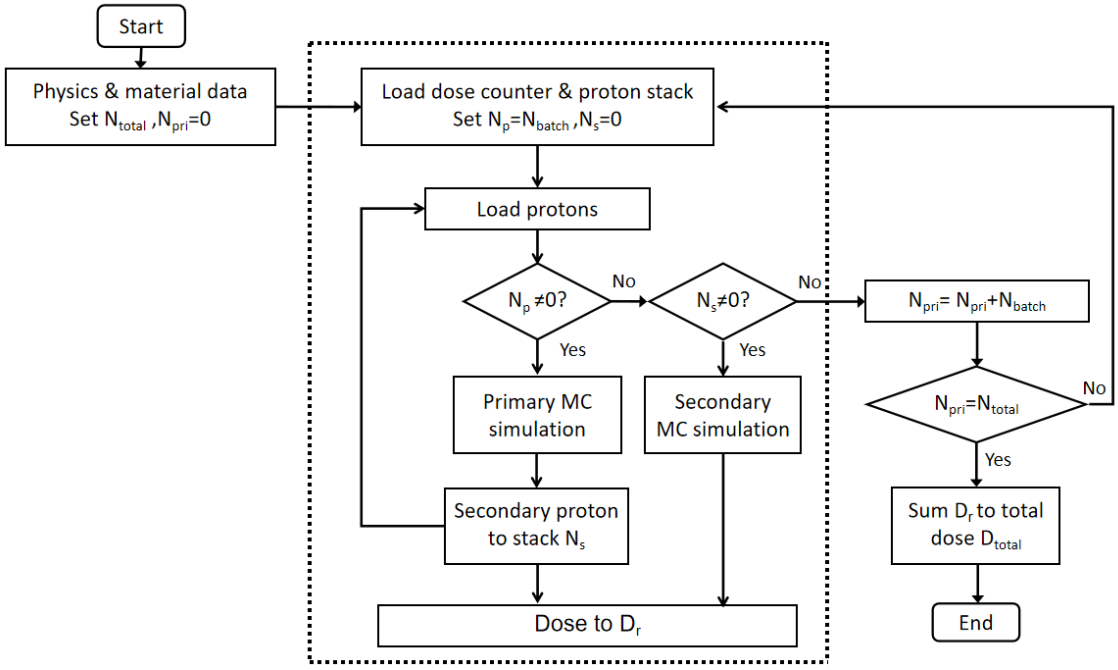


Figure 1: Schematic of the GPU acceleration of TPMC.

The GPU implementation of the aforementioned process is schematically illustrated in Figure 1. Protons were simulated in batches with a batch size of protons in all cases, except for the last batch in our simulation. After primary protons were loaded into a batch, the transport simulation was initiated. It is noteworthy that is equal to . In each batch, the dose information of primary protons, secondary protons, other secondary fragments, and all tertiary particles was calculated following the physics model and saved in the local dose counter . After all the particle transports were performed in a batch, the dose information in was sent to the CPU memory and added to the total dose, . The simulation process ends when the number of simulated primary protons is equal to the number of total primary protons .

In this study, all TPMC GPU dose calculation processes were performed on an NVIDIA Tesla V100 PCIe 32GB card system with 5120 processor cores. The operating frequency was 1230 MHz with

14teraFLOPS for the single-precision mode. All the CPU projects were performed on an Intel Xeon 8260 CPU with 24 processor cores system ,processor base frequency of 2.4 GHz, and 1 TB of maximum memory.

2.3 Test model

We used a Gaussian pencil beam with a spot size of and scattering angle of . The number of primary protons simulated was 10^7 and the beam energies were 110, 160, and 200 MeV. Four material phantoms containing water, lung tissue, and bone tissue were designed for testing, as shown in Figure 2. The size of the phantom was 10 cm × 10 cm × 30 cm, with a 1 mm × 1 mm×1.5 mm scoring grid. The densities and elemental compositions of the materials are listed in Table 2.

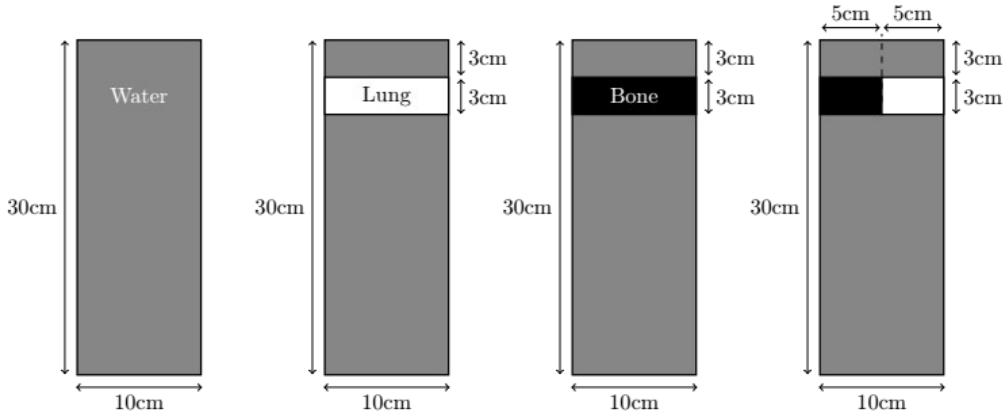


Figure 2: Size, shape, and materials of the four testing phantoms.

Table 2: Densities and elemental compositions of the materials in the testing phantoms. The densities and composition values are in units of g/cm^3 and mass weighted percentage, respectively.

Material	Density	H	C	N	O	Na	P	S	Cl	K	Mg	Ca
Water	1.0	11.2			88.8							
Lung	0.3	10.3	10.5	3.1	74.9	0.2	0.2	0.3	0.3	0.2		
Bone	1.9	13.5	16.0	4.2	44.5	0.3	9.5	0.3			0.2	21.5

3. Results

3.1 Dose validation

The physics setting of Topas included the following Geant4 modules: g4em-standard_opt3, g4hphy_QGSP_BIC_HP, g4decay, g4ion-binarycascade, g4helastic_HP, and g4q-stopping, following Testa et al.[27].

3.1.1 Depth dose curve

Figure 3 shows the depth-dose curves obtained for TPMC and Topas in the pure water phantom (phantom in Figure 2). The dose values for beams with different energies are normalized by the maximum dose (Bragg peak) of 110 MeV. The dose differences between TPMC and Topas mainly occur around the Bragg peak regions.

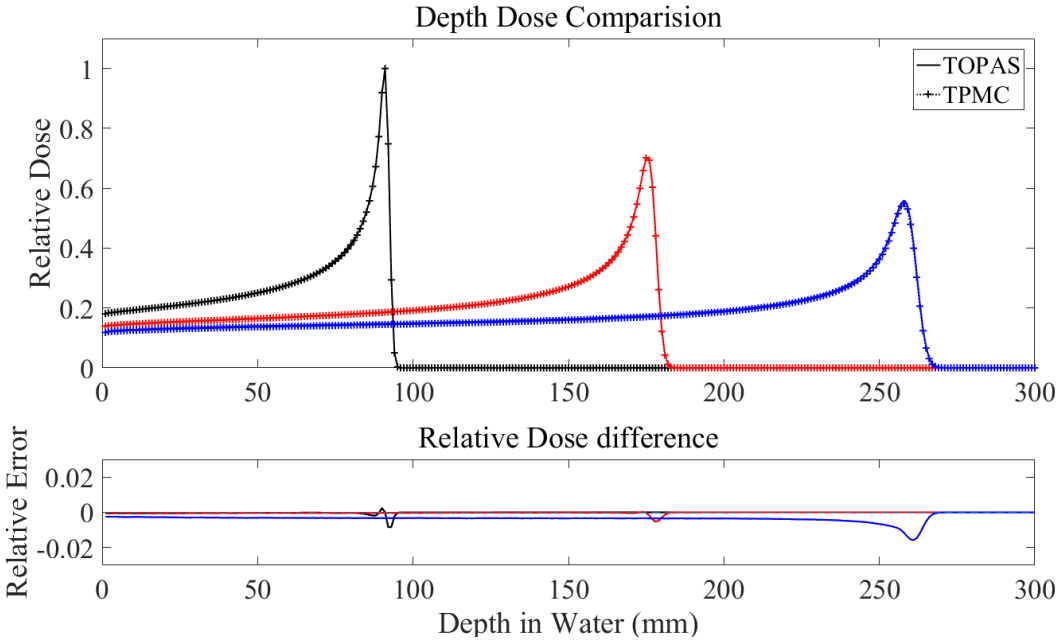


Figure 3: Depth dose distribution (top) and relative dose difference (bottom) for the pure water phantom for monoenergetic proton pencil beams of 110 MeV (black peak at 91 mm), 160 MeV (red peak at 175 mm), and 200 MeV (blue peak at 258 mm), calculated by TPMC and Topas. The dose value for different energy beams is normalized with respect to the maximum dose (Bragg peak) of 110 MeV.

3.1.2 Lateral dose verification

Lateral dose verification was performed by comparing the results for the four phantoms calculated by the TPMC and Topas at the same incident proton energy, E = 160 MeV. The relative error maps of the dose distributions are shown in Figure 4 (110MeV data normalized). In most of the regions with energy deposition, the results of TPMC are consistent with those of Topas. These differences mainly appear in the Bragg peak areas for phantoms (a), (b), and (c). Interestingly, for the pure water phantom (a), our predicted dose values are larger than those of Topas at energies before the Bragg peak, and TPMC decreases faster than Topas after the Bragg peak. These differences increase further when lung tissue is inserted into the proton beam pathway. However, the opposite results are obtained when the inserts were replaced with bone tissue, as shown in panels (b) and (c) of Figure 4. For the heterogeneous phantom, large differences appear along the extension line of the boundary between the two inserted materials, apart from the two Bragg

peaks. Despite this difference, the Bragg peak regions of TPMC and Topas are consistent in all the test cases.

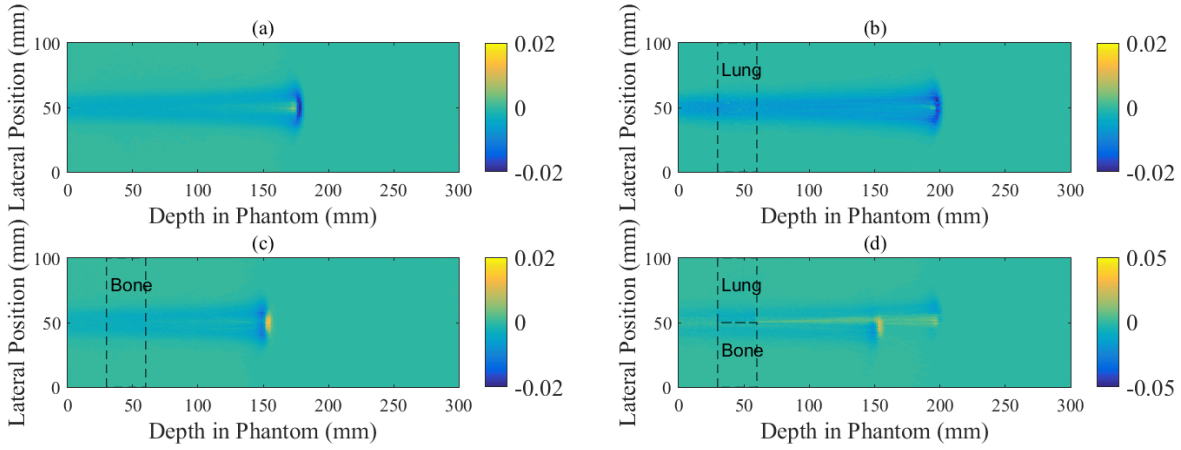
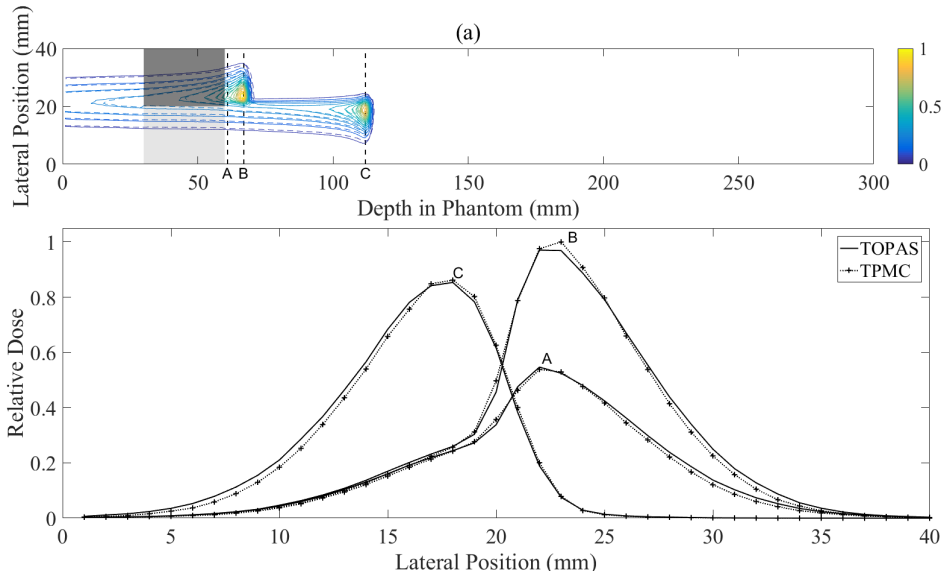


Figure 4: Lateral dose difference of TPMC and Topas in the four test phantoms for a 160 MeV monoenergetic proton pencil beam: (a) difference in water phantom, (b) difference in lung phantom, (c) difference in bone phantom, and (d) difference in heterogeneous phantom.

The influence of the heterogeneous medium was further studied by simulating two additional incident proton energies, 110 MeV and 200 MeV, for phantom (d). The lateral dose contour maps of the TPMC and Topas in the three cases, the corresponding lateral profiles at a depth of 6 cm (material boundary), and the two Bragg peaks are shown in Figures 5 (a-c). The results show that the lateral dose distribution in TPMC is consistent with that in Topas, regardless of the proton beam energy, phantom depth, or material difference.



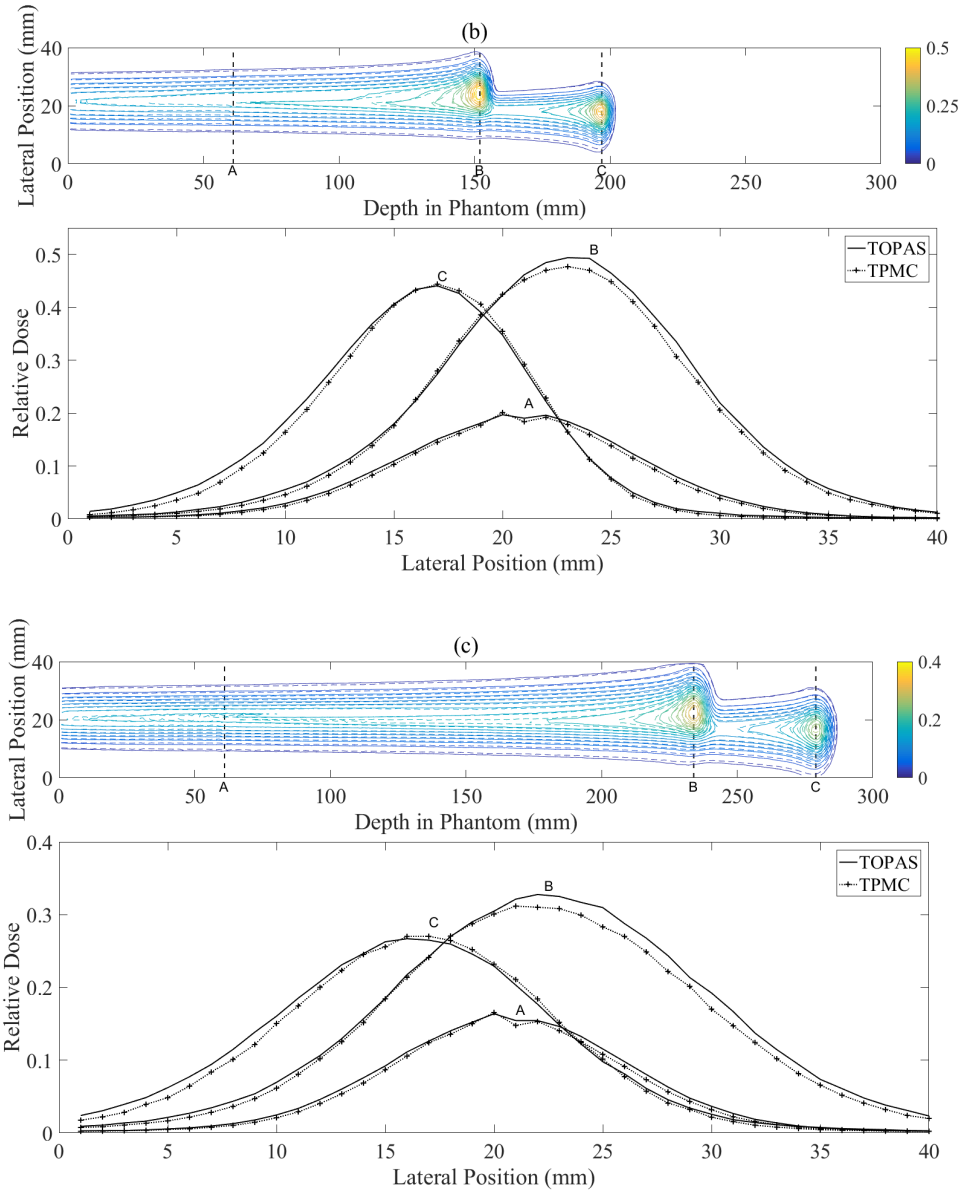


Figure 5: Lateral dose difference of TPMC and Topas in the heterogeneous phantoms for: (a) 110, (b) 160, and (c) 200 MeV mono-energetic proton pencil beams.

3.1.3 Quantitative analysis

To evaluate the accuracy of the TPMC quantitatively, we calculated the gamma passing rates for the four testing phantoms with three incident proton beam energies. The criteria were set to 1 mm/1% and 2 mm/2%. A mask was added to each phantom whose height and width were set to 5 cm and 30 cm, respectively. The depth of the mask was 1.1 times the depth of the Bragg peak. A deeper Bragg peak was selected for the heterogeneous phantom. The gamma passing rates in various situations are listed in Table 3. We note that all the values are greater than 0.99 and 0.95 under the criteria of 2 mm/2% and 1 mm/1%, respectively. These quantitative results indicate that the TPMC is sufficiently precise for proton treatment

planning. In addition, the gamma passing rates of the TPMC gradually decrease with increasing proton beam energy and complexity of the phantom.

Table 3: Masked Gamma passing rates of TPMC for four testing phantoms with 110, 160, and 200 MeV proton beams.

Energy	Phantom	Criteria	
		1mm/1%	2mm/2%
200 MeV	Water	99.2	99.9
	Water + Lung	99.0	99.9
	Water + Bone	98.0	99.9
	Heterogeneous	96.6	99.3
160 MeV	Water	99.1	99.9
	Water + Lung	98.7	99.7
	Water + Bone	98.1	99.9
	Heterogeneous	97.5	99.5
110 MeV	Water	99.4	99.9
	Water + Lung	99.6	99.9
	Water + Bone	98.3	99.6
	Heterogeneous	98.1	99.7

3.1.4 CT Image Validation

Next, we expanded our validation to a real patient CT phantom. A square beam with an energy of 110 MeV was incident. The dose distributions are shown in Figure 6. The TPMC dose calculation results for the horizontal and inclined injection beams are shown in Figures 6a and 6b, and the TOPAS results are plotted in Figures 6c and 6d. Figures 6e and 6f depict the dose differences between the two MC results. For regions with doses above 50% of the maximum dose, the average dose difference is less than 1.5%. We also performed a gamma-index rate test (2 mm/2%), and the passing rate exceeds 99%. Focusing on the dose points within the 10% isodose lines, the passing rate is greater than 96% for both cases.

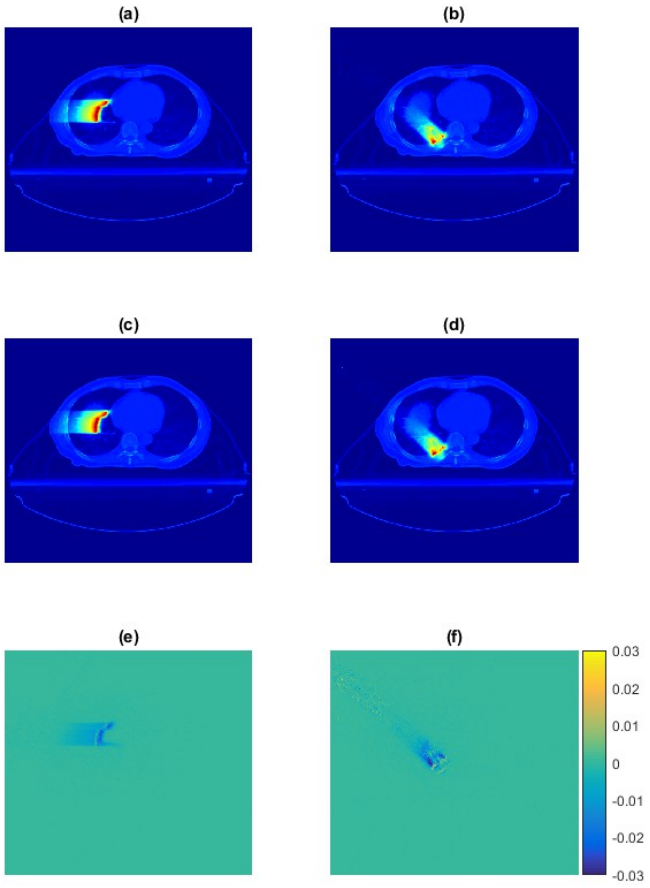


Figure 6: Dose calculation result in the case of the patient. The first two rows represent the dose results from TPMC and Topas respectively. The third row shows the relative dose difference between the two methods

3.2 Acceleration verification

The efficiency of the TPMC was tested on an NVIDIA Tesla V100 GPU card system and an Intel Xeon 8260 CPU processor (192 cores) system. The latter was also used in the full-MC simulation using Topas. We built the CUDA and OpenCL acceleration programs based on gPMC[10] and gPMC V2.0[14]. For comparison, the same physical model was also encoded on the CUDA and OpenCL platforms. In all the cases processed by GPUs, the local dose counter was optimized to eight to reduce memory conflicts as much as possible. The simulation times for the 10^7 incident protons for various beam energies, platforms, and devices are listed in Table 4. Clearly, the simulation time is primarily determined by the proton beam energies because high-energy protons have longer transport paths and induce more complicated nuclear reactions than low-energy protons. In addition, the simulation efficiency of the three GPU-based platforms was at the same level (within 10 s) and all met the clinical requirements. Owing to the acceleration of the GPUs and simplification of the physics model, the calculation time is generally approximately 7000 times faster than that of the full MC simulations by Topas. Taichi did not show explicit improvements in simulation time compared with CUDA and OpenCL because proton transport workflows are essentially the same in GPU-based platforms.

Table 4: Comparison of the dose simulation time (in seconds) for 10^7 incident protons with various beam energies, platforms and devices.

Energy	GPU (Nvidia Tesla V100)			CPU (Intel Xeon 8260)
	Taichi	OpenCL	CUDA	Topas
200 MeV	**9.00**	8.73	6.34	58890
160 MeV	**4.96**	6.92	4.96	38650
110 MeV	**3.34**	5.23	2.57	19560

3.3 Simplification of the code

As mentioned in the previous Section 1, Taichi integrates numerous function code instructions for ease of use. We compared the total number of lines of code written in Taichi, OpenCL, and CUDA, as shown in Table 5. The number of lines of code for programming the physics model was the same for all the three platforms. However, Taichi significantly reduced the total number of lines of code, which was approximately 11 times less than OpenCL and eight times less than CUDA. Such differences mainly come from the GPU implementation because it is quite easy in Taichi where only two lines of code are needed, compared to the thousands of lines needed in other platforms. This improvement makes TPMC an easy-to-use package that allows algorithm developers, students, and medical physicists to focus more on the underlying physics of proton transport rather than on code writing techniques.

Table 5: Comparison of the total number of lines of code and number of lines for the physics model in Taichi, OpenCL and CUDA.

	Taichi	OpenCL	CUDA
Total	**780**	~ 9000	~ 6000
Physics	**720**	~ 1000	~ 1000

4. Discussion

To ensure the accuracy of our physics model, it is worth noting that the maximum depth-dose difference in the pure water phantom occurred around the Bragg peak region. This is because nuclear reactions play a dominant role in the production of many short-range secondary fragments. However, the energy of these fragments was assumed to be locally deposited without additional interactions in the TPMC. This difference increases with increasing incident proton energy, which can be understood in the same way that the higher the projectiles energy, the more the fragments produced. In addition, the lateral dose difference mainly originates from the approximate treatment of the delta electrons, whose influence on the proton

transport directions is neglected in our physics model. The accuracy of TPMC decreased when other materials, such as lung or bone tissues, were inserted into water. The transport of delta electrons is probably non-negligible in low density materials for lung tissues [10], while for bone tissues, proton-calcium inelastic nuclear interactions should be included. Moreover, the phantom with the heterogeneous medium exhibited the lowest gamma passing rate. The cross section of protons and materials might dramatically change when protons pass through the heterogeneous interface, but we only consider the material information of the current voxel in that case. These results indicate that the accuracy of TPMC may decrease further when the structure of the human body is more complicated.

The lateral dose differences of TPMC and Topas in Figure 4 exhibit opposite results when lung or bone tissues were inserted compared to those of the pure water phantom. This was attributed to the material conversion method used to calculate the water-equivalent steps in Eq. (2). For lung and bone tissues, although energy loss is assumed to be conserved in such conversion, the value of is not absolutely accurate, since it is obtained by fitting to experimental data.[11] Thus, the energy of the protons after passing through the inserts (other than water) changed compared with the real situation, leading to a shift of approximately 1 mm in the Bragg peak position.

Although CUDA and OpenCL can achieve better comuputaion performance through numerous well developed functions.The implementation of Taichi in proton dose calculations appears to have overcome some drawbacks of CUDA and OpenCL because it supports multiple GPU devices and has a low programming barrier. In addition to these two benefits, Taichi also exhibits advantages in other aspects. In proton transport, Taichi continues to process all the particles on the GPUs, whereas OpenCL sends the information of the generated secondary protons from the GPU to the CPU memory after the simulation of each batch finishes. This information is returned to the GPU during secondary proton transport. Such cross-transportation may lead to a relatively low efficiency, as shown in Table 4, where the simulation time of OpenCL is slightly longer than that of Taichi. In addition, Taichi supports most Python function libraries such as Pytorch and Numpy, which naturally integrate dose calculations within the framework of deep learning. This feature provides a new aspect for future work, namely, using convolution neural networks to improve the accuracy of the present study. Similar studies were conducted by Ryan Neph et al. [20], who used 3D Unet and high-noise dose maps (10^5 incident particles) to predict low-noise dose distributions (10^7 incident particles) in treatment planning. Another work by Wu et al. [28] improved the accuracy of the pencil beam algorithm using Unet. Both studies yielded promising results, indicating that deep learning has great potential for dose prediction. Finally, the great power of Taichi in gradient descent optimization can be applied to other aspects, such as image processing and dose optimization. Our goal is to develop a Taichi-based treatment planning system for proton therapy in the future.

5. Conclusion

We developed a graphics processing unit (GPU)-based Monte Carlo proton dose calculation engine within the Taichi framework. Protons are transported in a class II condensed history simulation scheme with various post step proton-nucleus interactions. The simulations were parallelized on GPUs for acceleration. The results of the testing models indicated that the accuracy satisfied the clinical requirements adequately. The simulation speed was approximately 7000 times faster than that of the full MC simulation using Topas, and the number of lines of code was approximately 10 times lesser than those of CUDA and OpenCL. Our study provides a fast, accurate, and easy-to-use proton dose calculation engine for algorithm developers, students, and medical physicists. Further studies are needed to implement Taichi in other components of treatment planning, such as image processing and dose optimization.

References

[1] F.Aliyah, SG.Pinasti, AA.Rahman. Proton therapy facilities: an overview of the development in recent year. IOP Conf. Ser: Earth Environ Sci. 927,012042 (2021). doi:10.1088/1755-1315/927/1/012042

[2]B.Qin, X.Liu, Q.S.Chen et al., Design and development of the beamline for a proton therapy system. NUCL SCI TECH 32, 138 (2021).doi: 10.1007/s41365-021-00975-y

[3]W.C Fang, X.X Huang,J.H Tan et al., Proton linac-based therapy facility for ultra-high dose rate (FLASH) treatment. NUCL SCI TECH 32, 34 (2021). doi: 10.1007/s41365-021-00872-4

[4]J.Saini, E.Traneus, D.Maes et al., Advanced Proton Beam Dosimetry Part I: review and performance evaluation of dose calculation algorithms. Transl Lung Cancer Res. 7(2):171-179 (2018). doi:10.21037/tlcr.2018.04.05

[5]S.Muraro, G.Battistoni, AC.Kraan. Challenges in Monte Carlo simulations as clinical and research tool in particle therapy: a review. Front. Phys. 8:567800 (2020). doi:10.3389/fphy.2020.567800

[6]H.Paganetti. Relative biological effectiveness (RBE) values for proton beam therapy. Variations as a function of biological endpoint, dose, and linear energy transfer. Phys. Med. Biol. 59,R419 (2014). doi:10.1088/0031-9155/59/22/R419

[7]A.K Hu, R Qiu, H Liu et al., THUBrachy: fast Monte Carlo dose calculation tool accelerated by heterogeneous hardware for high-dose-rate brachytherapy. NUCL SCI TECH 32, 32 (2021). doi:10.1007/s41365-021-00866-2

[8]F.Liu, N.Jansson, A.Podobas et al., Accelerating Radiation Therapy Dose Calculation with Nvidia GPUs. Paper presented at the 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (Portland, OR, USA). doi:10.1109/IPDPSW52791.2021.00076

[9]J.Gajewski, M.Garbacz, C.W.Chang et al., Commissioning of GPU–accelerated monte carlo code FRED for clinical applications in proton therapy. Front. Phys. 8:567300 (2021). doi:10.3389/fphy.2020.567300

[10]X.Jia, J.Schümann, H.Paganetti et al. GPU-based fast Monte Carlo dose calculation for proton therapy. Phys. Med. Biol. 57,7783 (2012). doi:10.1088/0031-9155/57/23/7783

[11]M.Fippel, M.Soukup. A Monte Carlo dose calculation algorithm for proton therapy. Med Phys. 31(8),2263-73 (2004). doi:10.1118/1.1769631

[12]H.Wan.Chan.Tseung, J.Ma, C.Beltran. A fast GPU-based Monte Carlo simulation of proton transport with detailed modeling of nonelastic interactions. Med Phys. 42(6),2967-78 (2015). doi:10.1118/1.4921046

[13]Z.Tian, F.Shi, M.Folkerts et al., A GPU OpenCL based cross-platform Monte Carlo dose calculation engine (goMC). Phys. Med. Bio. 60,7419 (2015). doi:10.1088/0031-9155/60/19/7419

[14]N.Qin, P.Botas, D.Giantsoudi et al., Recent developments and comprehensive evaluations of a GPU-based Monte Carlo package for proton therapy. Phys. Med. Bio. 61,7347 (2016). doi:10.1088/0031-9155/61/20/7347

[15]Y.Hu, T.M.Li, L.Anderson et al., Taichi: a language for high-performance computation on spatially sparse data structure. ACM Trans. Graph. 38,6, Article 201 (2019). doi:10.1145/3355089.3356506

[16]Y.Hu, L.Anderson, T.M.Li et al., Difftaichi: Differentiable programming for physical simulation. Paper presented at the 2020 International Conference on Learning Representations(Virtual Conference, Formerly Addis Ababa ETHIOPIA 2020).

[17]Y.Hu, J.Liu, X.Yang, et al., Quantaichi: a compiler for quantized simulations. ACM Trans. Graph. 40, 4, Article 182 (2021). doi:10.1145/3450626.3459671

[18]C.Kontaxis, GH.Bol, JJW.Lagendijk et al., DeepDose: towards a fast dose calculation engine for radiation therapy using deep learning. Phys. Med. Bio. 65,075013 (2020). doi:10.1088/1361-6560/ab7630

[19]Y.Xing, D.Nguyen, W.Lu et al., A feasibility study on deep learning-based radiotherapy dose calculation. Med Phys. 47(2), 753-758 (2020). doi:10.1002/mp.13953

[20]R.Neph, Q.Lyu, Y.Huang et al., DeepMC: a deep learning method for efficient Monte Carlo beamlet dose calculation by predictive denoising in magnetic resonance-guided radiotherapy. Phys. Med. Bio. 66,035022 (2021). doi:10.1088/1361-6560/abca01

[21]J.Liang, MC.Lin . Machine learning for digital try-on: Challenges and progress. Comp Visual Media.7,159–167(2021). doi:10.1007/s41095-020-0189-1

[22]I.Kawrakow. Accurate condensed history Monte Carlo simulation of electron transport.I.EGSnrc,the new EGS4 version. Med Phys. 27(3),485-98 (2000). doi:10.1118/1.598917

[23]WD.Newhauser, R.Zhang. The physics of proton therapy. Phys. Med. Bio. 60,R155 (2015). doi:10.1088/0031-9155/60/8/R155

[24]JF.Ziegler. Comments on icru report no. 49: stopping powers and ranges for protons and alpha particles. Radiat Res. 152(2), 219–222 (1999). doi:10.2307/3580097

[25]Grimes, D.Robert, D.Warren. An approximate analytical solution of the Bethe equation for charged particles in the radiotherapeutic energy range. Sci. Rep. 7, 9781 (2017). doi:10.1038/s41598-017-10554-0

[26]J. Allison; K. Amako; J. Apostolakis et al., Geant4 developments and applications,IEEE T NUCL SCI, 53, n01(2006), doi: 10.1109/TNS.2006.869826.

[27]M. Testa, J.Schümann, H.M.Lu et al., Experimental validation of the TOPAS Monte Carlo system for passive scattering proton therapy. Med Phys. 40(12),121719 (2013). doi:10.1118/1.4828781

[28]C.Wu, D.Nguyen, Y.Xing et al. Improving proton dose calculation accuracy by using deep learning. Mach Learn Sci Technol. 2(1),015017 (2021). doi:10.1088/2632-2153/abb6d5